PCT WORLD INTELLECTUAL PROPERTY ORGANIZATION
International Bureau

INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(51) International Patent Classificati n 6 : G07F 7/10, H04L 9/32 — A1

(11) International Publication Number: WO 97/41539

(43) International Publication Date: 6 November 1997 (06.11.97)

(21) International Application Number: PCT/US97/06934

(22) International Filing Date: 24 April 1997 (24.04.97)

(30) Priority Data:
08/639,909 26 April 1996 (26.04.96) US

(71) Applicant *(for all designated States except US)*: VERIFONE, INC. [US/US]; Suite 400, Three Lagoon Drive, Redwood City, CA 94065 (US).

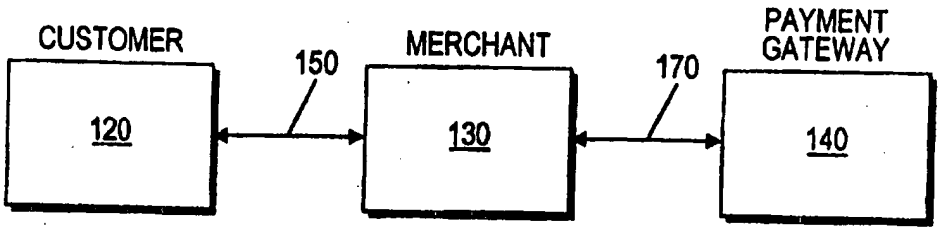
(72) Inventor; and
(75) Inventor/Applicant *(for US only)*: ROWNEY, Kevin, T., B. [US/US]; 748 Duncan Street, San Francisco, CA 94131 (US).

(74) Agents: STEPHENS, L., Keith; Cooley Godward LLP, 3000 El Camino Real, Five Palo Alto Square, Palo Alto, CA 94306-2155 (US) et al.

(81) Designated States: AL, AM, AT, AU, AZ, BA, BB, BG, BR, BY, CA, CH, CN, CU, CZ, DE, DK, EE, ES, FI, GB, GE, GH, HU, IL, IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MD, MG, MK, MN, MW, MX, NO, NZ, PL, PT, RO, RU, SD, SE, SG, SI, SK, TJ, TM, TR, TT, UA, UG, US, UZ, VN, YU, ARIPO patent (GH, KE, LS, MW, SD, SZ, UG), Eurasian patent (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European patent (AT, BE, CH, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE), OAPI patent (BF, BJ, CF, CG, CI, CM, GA, GN, ML, MR, NE, SN, TD, TG).

Published
With international search report.
Before the expiration of the time limit for amending the claims and to be republished in the event of the receipt of amendments.

(54) Title: A SYSTEM AND METHOD FOR SECURE NETWORK ELECTRONIC PAYMENT AND CREDIT COLLECTION



(57) Abstract

Secure transmission of data is provided between a plurality of computer systems over a public communication system, such as the Internet. Secure transmission of data is provided from a customer computer system to a merchant computer system, and for the further secure transmission of data from the merchant computer system to a payment gateway computer system. The payment gateway system evaluates the information and returns authorization or denial of credit via a secure transmission to the merchant which is communicated to the customer by the merchant.

FOR THE PURPOSES OF INFORMATION ONLY

Codes used to identify States party to the PCT on the front pages of pamphlets publishing international applications under the PCT.

AL	Albania
AM	Armenia
AT	Austria
AU	Australia
AZ	Azerbaijan
BA	Bosnia and Herzegovina
BB	Barbados
BE	Belgium
BF	Burkina Faso
BG	Bulgaria
BJ	Benin
BR	Brazil
BY	Belarus
CA	Canada
CF	Central African Republic
CG	Congo
CH	Switzerland
CI	Côte d'Ivoire
CM	Cameroon
CN	China
CU	Cuba
CZ	Czech Republic
DE	Germany
DK	Denmark
EE	Estonia
ES	Spain
FI	Finland
FR	France
GA	Gabon
GB	United Kingdom
GE	Georgia
GH	Ghana
GN	Guinea
GR	Greece
HU	Hungary
IE	Ireland
IL	Israel
IS	Iceland
IT	Italy
JP	Japan
KE	Kenya
KG	Kyrgyzstan
KP	Democratic People's Republic of Korea
KR	Republic of Korea
KZ	Kazakstan
LC	Saint Lucia
LI	Liechtenstein
LK	Sri Lanka
LR	Liberia
LS	Lesotho
LT	Lithuania
LU	Luxembourg
LV	Latvia
MC	Monaco
MD	Republic of Moldova
MG	Madagascar
MK	The former Yugoslav Republic of Macedonia
ML	Mali
MN	Mongolia
MR	Mauritania
MW	Malawi
MX	Mexico
NE	Niger
NL	Netherlands
NO	Norway
NZ	New Zealand
PL	Poland
PT	Portugal
RO	Romania
RU	Russian Federation
SD	Sudan
SE	Sweden
SG	Singapore
SI	Slovenia
SK	Slovakia
SN	Senegal
SZ	Swaziland
TD	Chad
TG	Togo
TJ	Tajikistan
TM	Turkmenistan
TR	Turkey
TT	Trinidad and Tobago
UA	Ukraine
UG	Uganda
US	United States of America
UZ	Uzbekistan
VN	Viet Nam
YU	Yugoslavia
ZW	Zimbabwe

A SYSTEM AND METHOD FOR SECURE NETWORK ELECTRONIC PAYMENT AND CREDIT COLLECTION

Field Of The Invention

The present invention relates to the electronic payment in exchange for goods and services purchased over a communication network, and more specifically, and more particularly, to a system, method and article of manufacture for securely transmitting payment information from a customer to a merchant to a payment gateway and returning appropriate, secure authorization to the merchant and the customer.

Background of the Invention

It is desirable for a computer operated under the control of a merchant to obtain information offered by a customer and transmitted by a computer operating under the control of the customer over a publicly accessible packet-switched network (e.g., the Internet) to the computer operating under the control of the merchant, without risking the exposure of the information to interception by third parties that have access to the network, and to assure that the information is from an authentic source. It is further desirable to have the ability for the merchant to transmit information, including a subset of the information provided by the customer, over such a network to a payment gateway computer system that is authorized, by a bank or other financial institution that has the responsibility of providing payment on behalf of the customer, to authorize a commercial transaction on behalf of such a financial institution, without the risk of exposing that information to interception by third parties. Such institutions include, for example, financial institutions offering credit or debit card services.

One such attempt to provide such a secure transmission channel is a secure payment technology such as Secure Electronic Transaction (hereinafter "SET"), jointly developed by the Visa and MasterCard card associations, and described in Visa and MasterCard's *Secure Electronic Transaction (SET) Specification*, February 23, 1996, hereby incorporated by reference. Other such secure payment technologies include Secure Transaction Technology ("STT"), Secure Electronic Payments Protocol ("SEPP"), Internet Keyed Payments ("iKP"), Net Trust, and Cybercash Credit Payment Protocol. One of ordinary skill in the art will readily comprehend that any of the secure payment technologies can be substituted for the SET protocol without undue experimentation. Such secure payment technologies require the customer to operate software that is compliant with the secure payment technology, interacting with third-party certification authorities, thereby allowing the customer to transmit encoded information to a merchant, some of which may be decoded by the merchant, and some which can be decoded only by a payment gateway specified by the customer. A drawback to the secure payment technology approach is that it requires deployment of special-purpose software compliant with the particular secure payment technology to the customer, thereby limiting user acceptance of the secure payment technology to those customers willing to install that software. Customers are generally reluctant to install such specialized software in the absence of a general acceptance of merchant software and payment gateway software that incorporate the corresponding secure payment technology with which to interact. Similarly, merchants and payment gateways are reluctant to implement a secure payment technology in the absence of an installed customer base that is available to use that secure payment technology. This presents a "chicken-and-the-egg" problem in that no particular component of a

secure payment technology is likely to achieve general acceptance until the other components also achieve general acceptance.

Another such attempt to provide such a secure transmission channel is a general-purpose secure communication protocol such as Netscape, Inc.'s Secure Sockets Layer (hereinafter "SSL") , as described in Freier, Karlton & Kocher (hereinafter "Freier"), *The SSL Protocol Version 3.0*, March 1996, and hereby incorporated by reference. SSL provides a means for secure transmission between two computers. SSL has the advantage that it does not require special-purpose software to be installed on the customer's computer because it is already incorporated into widely available software that many people utilize as their standard Internet access medium, and does not require that the customer interact with any third-party certification authority. Instead, the support for SSL may be incorporated into software already in use by the customer, e.g., the Netscape Navigator World Wide Web browsing tool. However, although a computer on an SSL connection may initiate a second SSL connection to another computer, a drawback to the SSL approach is each SSL connection supports only a two-computer connection. Therefore, SSL does not provide a mechanism for transmitting encoded information to a merchant for retransmission to a payment gateway such that a subset of the information is readable to the payment gateway but not to the merchant. Although SSL allows for robustly secure two-party data transmission, it does not meet the ultimate need of the electronic commerce market for robustly secure three-party data transmission. Other examples of general-purpose secure communication protocols include Private Communications Technology ("PCT") from Microsoft, Inc., Secure Hyper-Text Transport Protocol ("SHTTP") from Theresa Systems, Shen, Kerberos, Photuris, Pretty Good Privacy ("PGP") and

Ipv6 which meets the IPSEC criteria. One of ordinary skill in the art will readily comprehend that any of the general-purpose secure communication protocols can be substituted for the SSL transmission protocol without undue experimentation.

OBJECTS OF THE INVENTION

It is desirable to provide a hybrid approach that encourages the deployment of a three-party secure channel such as SET by payment gateways in the absence of customer acceptance, thereby providing customers with an incentive to install SET-compliant software on their computer systems. It is further desirable to provide a means by which a merchant may communicate with a customer using a readily deployed secure channel such SSL or another general-purpose secure communication protocol, and communicate with a payment gateway using a modified SET-like protocol that is not dependent upon customer certification.

SUMMARY OF THE INVENTION

According to a broad aspect of a preferred embodiment of the invention, secure transmission of data is provided between a plurality of computer systems over a public communication system, such as the Internet. Secure transmission of data is provided from a customer computer system to a merchant computer system, and for the further secure transmission of data from the merchant computer system to a payment gateway computer system. The payment gateway system evaluates the information and returns authorization or denial of credit via a secure transmission to the merchant which is communicated to the customer by the merchant.

DESCRIPTION OF THE DRAWINGS

The foregoing and other objects, aspects and advantages are better understood from the following detailed description of a preferred embodiment of the invention with reference to the drawings, in which:

Figure **1A** is a block diagram of a representative hardware environment in accordance with a preferred embodiment;

Figure **1B** depicts an overview in accordance with a preferred embodiment;

Figure **2** depicts a more detailed view of a customer computer system in communication with merchant system under the Secure Sockets Layer protocol in accordance with a preferred embodiment;

Figure **3** depicts an overview of the method of securely supplying payment information to a payment gateway in order to obtain payment authorization in accordance with a preferred embodiment;

Figure **4** depicts the detailed steps of generating and transmitting a payment authorization request in accordance with a preferred embodiment;

Figures **5A** through **5F** depict views of the payment authorization request and its component parts in accordance with a preferred embodiment;

Figures **6A** and **6B** depict the detailed steps of processing a payment authorization request and generating and transmitting a payment

authorization request response in accordance with a preferred embodiment;

Figures **7A** through **7J** depict views of the payment authorization response and its component parts in accordance with a preferred embodiment;

Figure **8** depicts the detailed steps of processing a payment authorization response in accordance with a preferred embodiment;

Figure **9** depicts an overview of the method of securely supplying payment capture information to a payment gateway in accordance with a preferred embodiment;

Figure **10** depicts the detailed steps of generating and transmitting a payment capture request in accordance with a preferred embodiment;

Figures **11A** through **11F** depict views of the payment capture request and its component parts in accordance with a preferred embodiment;

Figures **12A** and **12B** depict the detailed steps of processing a payment capture request and generating and transmitting a payment capture request response in accordance with a preferred embodiment;

Figures **13A** through **13F** depict views of the payment capture response and its component parts in accordance with a preferred embodiment; and

Figure **14** depicts the detailed steps of processing a payment capture response in accordance with a preferred embodiment.

DETAILED DESCRIPTION

A preferred embodiment of a system in accordance with the present invention is preferably practiced in the context of a personal computer such as the IBM PS/2, Apple Macintosh computer or UNIX based workstation. A representative hardware environment is depicted in Figure **1A**, which illustrates a typical hardware configuration of a workstation in accordance with a preferred embodiment having a central processing unit **10**, such as a microprocessor, and a number of other units interconnected via a system bus **12**. The workstation shown in Figure **1A** includes a Random Access Memory (RAM) **14**, Read Only Memory (ROM) **16**, an I/O adapter **18** for connecting peripheral devices such as disk storage units **20** to the bus **12**, a user interface adapter **22** for connecting a keyboard **24**, a mouse **26**, a speaker **28**, a microphone **32**, and/or other user interface devices such as a touch screen (not shown) to the bus **12**, communication adapter **34** for connecting the workstation to a communication network (e.g., a data processing network) and a display adapter **36** for connecting the bus **12**
to a display device **38**. The workstation typically has resident thereon an operating system such as the Microsoft Windows Operating System (OS), the IBM OS/2 operating system, the MAC OS, or UNIX operating

system. Those skilled in the art will appreciate that the present invention may also be implemented on platforms and operating systems other than those mentioned.

A preferred embodiment is written using JAVA, C, and the C++ language and utilizes object oriented programming methodology. Object oriented programming (OOP) has become increasingly used to develop complex applications. As OOP moves toward the mainstream of software design and development, various software solutions will need to be adapted to make use of the benefits of OOP. A need exists for these principles of OOP to be applied to a messaging interface of an electronic messaging system such that a set of OOP classes and objects for the messaging interface can be provided.

OOP is a process of developing computer software using objects, including the steps of analyzing the problem, designing the system, and constructing the program. An object is a software package that contains both data and a collection of related structures and procedures. Since it contains both data and a collection of structures and procedures, it can be visualized as a self-sufficient component that does not require other additional structures, procedures or data to perform its specific task. OOP, therefore, views a computer program as a collection of largely autonomous components, called objects, each of which is responsible for a specific task. This concept of packaging data, structures, and procedures together in one component or module is called encapsulation.

In general, OOP components are reusable software modules which present an interface that conforms to an object model and which are accessed at run-time through a component integration architecture.

A component integration architecture is a set of architecture mechanisms which allow software modules in different process spaces to utilize each others capabilities or functions. This is generally done by assuming a common component object model on which to build the architecture.

It is worthwhile to differentiate between an object and a class of objects at this point. An object is a single instance of the class of objects, which is often just called a class. A class of objects can be viewed as a blueprint, from which many objects can be formed.

OOP allows the programmer to create an object that is a part of another object. For example, the object representing a piston engine is said to have a composition-relationship with the object representing a piston. In reality, a piston engine comprises a piston, valves and many other components; the fact that a piston is an element of a piston engine can be logically and semantically represented in OOP by two objects.

OOP also allows creation of an object that "depends from" another object. If there are two objects, one representing a piston engine and the other representing a piston engine wherein the piston is made of ceramic, then the relationship between the two objects is not that of composition. A ceramic piston engine does not make up a piston engine. Rather it is merely one kind of piston engine that has one more limitation than the piston engine; its piston is made of ceramic. In this case, the object representing the ceramic piston engine is called a derived object, and it inherits all of the aspects of the object representing the piston engine and adds further limitation or detail to it. The object representing the ceramic piston engine "depends from"

the object representing the piston engine. The relationship between these objects is called inheritance.

When the object or class representing the ceramic piston engine inherits all of the aspects of the objects representing the piston engine, it inherits the thermal characteristics of a standard piston defined in the piston engine class. However, the ceramic piston engine object overrides these ceramic specific thermal characteristics, which are typically different from those associated with a metal piston. It skips over the original and uses new functions related to ceramic pistons. Different kinds of piston engines will have different characteristics, but may have the same underlying functions associated with it (e.g., how many pistons in the engine, ignition sequences, lubrication, etc.). To access each of these functions in any piston engine object, a programmer would call the same functions with the same names, but each type of piston engine may have different/overriding implementations of functions behind the same name. This ability to hide different implementations of a function behind the same name is called polymorphism and it greatly simplifies communication among objects.

With the concepts of composition-relationship, encapsulation, inheritance and polymorphism, an object can represent just about anything in the real world. In fact, our logical perception of the reality is the only limit on determining the kinds of things that can become objects in object-oriented software. Some typical categories are as follows:

- Objects can represent physical objects, such as automobiles in a traffic-flow simulation, electrical components in a

circuit-design program, countries in an economics model, or aircraft in an air-traffic-control system.

- Objects can represent elements of the computer-user environment such as windows, menus or graphics objects.
- An object can represent an inventory, such as a personnel file or a table of the latitudes and longitudes of cities.
- An object can represent user-defined data types such as time, angles, and complex numbers, or points on the plane.

With this enormous capability of an object to represent just about any logically separable matters, OOP allows the software developer to design and implement a computer program that is a model of some aspects of reality, whether that reality is a physical entity, a process, a system, or a composition of matter. Since the object can represent anything, the software developer can create an object which can be used as a component in a larger software project in the future.

If 90% of a new OOP software program consists of proven, existing components made from preexisting reusable objects, then only the remaining 10% of the new software project has to be written and tested from scratch. Since 90% already came from an inventory of extensively tested reusable objects, the potential domain from which an error could originate is 10% of the program. As a result, OOP enables software developers to build objects out of other, previously built, objects.

This process closely resembles complex machinery being built out of assemblies and sub-assemblies. OOP technology, therefore, makes software engineering more like hardware engineering in that software is built from existing components, which are available to the developer

as objects. All this adds up to an improved quality of the software as well as an increased speed of its development.

Programming languages are beginning to fully support the OOP principles, such as encapsulation, inheritance, polymorphism, and composition-relationship. With the advent of the C++ language, many commercial software developers have embraced OOP. C++ is an OOP language that offers a fast, machine-executable code. Furthermore, C++ is suitable for both commercial-application and systems-programming projects. For now, C++ appears to be the most popular choice among many OOP programmers, but there is a host of other OOP languages, such as Smalltalk, common lisp object system (CLOS), and Eiffel. Additionally, OOP capabilities are being added to more traditional popular computer programming languages such as Pascal.

The benefits of object classes can be summarized, as follows:

- *Objects* and their corresponding classes break down complex programming problems into many smaller, simpler problems.
- *Encapsulation* enforces data abstraction through the organization of data into small, independent objects that can communicate with each other. Encapsulation protects the data in an object from accidental damage, but allows other objects to interact with that data by calling the object's member functions and structures.
- *Subclassing* and inheritance make it possible to extend and modify objects through deriving new kinds of objects from the standard classes available in the system. Thus, new capabilities are created without having to start from scratch.

- *Polymorphism* and multiple inheritance make it possible for different programmers to mix and match characteristics of many different classes and create specialized objects that can still work with related objects in predictable ways.
- *Class hierarchies* and containment hierarchies provide a flexible mechanism for modeling real-world objects and the relationships among them.
- *Libraries* of reusable classes are useful in many situations, but they also have some limitations. For example:
- *Complexity.* In a complex system, the class hierarchies for related classes can become extremely confusing, with many dozens or even hundreds of classes.
- *Flow of control.* A program written with the aid of class libraries is still responsible for the flow of control (i.e., it must control the interactions among all the objects created from a particular library). The programmer has to decide which functions to call at what times for which kinds of objects.
- *Duplication of effort.* Although class libraries allow programmers to use and reuse many small pieces of code, each programmer puts those pieces together in a different way. Two different programmers can use the same set of class libraries to write two programs that do exactly the same thing but whose internal structure (i.e., design) may be quite different, depending on hundreds of small decisions each programmer makes along the way. Inevitably, similar pieces of code end up doing similar things in slightly different ways and do not work as well together as they should.

Class libraries are very flexible. As programs grow more complex, more programmers are forced to reinvent basic solutions to basic problems over and over again. A relatively new extension of the class

library concept is to have a framework of class libraries. This framework is more complex and consists of significant collections of collaborating classes that capture both the small scale patterns and major mechanisms that implement the common requirements and design in a specific application domain. They were first developed to free application programmers from the chores involved in displaying menus, windows, dialog boxes, and other standard user interface elements for personal computers.

Frameworks also represent a change in the way programmers think about the interaction between the code they write and code written by others. In the early days of procedural programming, the programmer called libraries provided by the operating system to perform certain tasks, but basically the program executed down the page from start to finish, and the programmer was solely responsible for the flow of control. This was appropriate for printing out paychecks, calculating a mathematical table, or solving other problems with a program that executed in just one way.

The development of graphical user interfaces began to turn this procedural programming arrangement inside out. These interfaces allow the user, rather than program logic, to drive the program and decide when certain actions should be performed. Today, most personal computer software accomplishes this by means of an event loop which monitors the mouse, keyboard, and other sources of external events and calls the appropriate parts of the programmer's code according to actions that the user performs. The programmer no longer determines the order in which events occur. Instead, a program is divided into separate pieces that are called at unpredictable times and in an unpredictable order. By relinquishing

control in this way to users, the developer creates a program that is much easier to use. Nevertheless, individual pieces of the program written by the developer still call libraries provided by the operating system to accomplish certain tasks, and the programmer must still determine the flow of control within each piece after it's called by the event loop. Application code still "sits on top of" the system.

Even event loop programs require programmers to write a lot of code that should not need to be written separately for every application. The concept of an application framework carries the event loop concept further. Instead of dealing with all the nuts and bolts of constructing basic menus, windows, and dialog boxes and then making these things all work together, programmers using application frameworks start with working application code and basic user interface elements in place. Subsequently, they build from there by replacing some of the generic capabilities of the framework with the specific capabilities of the intended application.

Application frameworks reduce the total amount of code that a programmer has to write from scratch. However, because the framework is really a generic application that displays windows, supports copy and paste, and so on, the programmer can also relinquish control to a greater degree than event loop programs permit. The framework code takes care of almost all event handling and flow of control, and the programmer's code is called only when the framework needs it (e.g., to create or manipulate a proprietary data structure).

A programmer writing a framework program not only relinquishes control to the user (as is also true for event loop programs), but also

relinquishes the detailed flow of control within the program to the framework. This approach allows the creation of more complex systems that work together in interesting ways, as opposed to isolated programs, having custom code, being created over and over again for similar problems.

Thus, as is explained above, a framework basically is a collection of cooperating classes that make up a reusable design solution for a given problem domain. It typically includes objects that provide default behavior (e.g., for menus and windows), and programmers use it by inheriting some of that default behavior and overriding other behavior so that the framework calls application code at the appropriate times.

There are three main differences between frameworks and class libraries:

- *Behavior versus protocol.* Class libraries are essentially collections of behaviors that you can call when you want those individual behaviors in your program. A framework, on the other hand, provides not only behavior but also the protocol or set of rules that govern the ways in which behaviors can be combined, including rules for what a programmer is supposed to provide versus what the framework provides.
- *Call versus override.* With a class library, the code the programmer instantiates objects and calls their member functions. It's possible to instantiate and call objects in the same way with a framework (i.e., to treat the framework as a class library), but to take full advantage of a framework's reusable design, a programmer typically writes code that overrides and is called by the framework. The framework manages the flow of control among its objects. Writing a program involves dividing responsibilities among the various pieces

of software that are called by the framework rather than specifying how the different pieces should work together.

- *Implementation versus design.* With class libraries, programmers reuse only implementations, whereas with frameworks, they reuse design. A framework embodies the way a family of related programs or pieces of software work. It represents a generic design solution that can be adapted to a variety of specific problems in a given domain. For example, a single framework can embody the way a user interface works, even though two different user interfaces created with the same framework might solve quite different interface problems.

Thus, through the development of frameworks for solutions to various problems and programming tasks, significant reductions in the design and development effort for software can be achieved. A preferred embodiment of the invention utilizes HyperText Markup Language (HTML) to implement documents on the Internet together with a general-purpose secure communication protocol for a transport medium between the client and the merchant. HTML is a simple data format used to create hypertext documents that are portable from one platform to another. HTML documents are SGML documents with generic semantics that are appropriate for representing information from a wide range of domains. HTML has been in use by the World-Wide Web global information initiative since 1990. HTML is an application of ISO Standard 8879:1986 Information Processing Text and Office Systems; Standard Generalized Markup Language (SGML).

To date, Web development tools have been limited in their ability to create dynamic Web applications which span from client to server and interoperate with existing computing resources. Until recently, HTML

has been the dominant technology used in development of Web-based solutions. However, HTML has proven to be inadequate in the following areas:

- o Poor performance;
- o Restricted user interface capabilities;
- o Can only produce static Web pages;
- o Lack of interoperability with existing applications and data; and
- o Inability to scale.

Sun Microsystem's Java language solves many of the client-side problems by:

- o Improving performance on the client side;
- o Enabling the creation of dynamic, real-time Web applications; and
- o Providing the ability to create a wide variety of user interface components.

With Java, developers can create robust User Interface (UI) components. Custom "widgets" (e.g. real-time stock tickers, animated icons, etc.) can be created, and client-side performance is improved. Unlike HTML, Java supports the notion of client-side validation, offloading appropriate processing onto the client for improved performance. Dynamic, real-time Web pages can be created. Using the above-mentioned custom UI components, dynamic Web pages can also be created.

Sun's Java language has emerged as an industry-recognized language for "programming the Internet." Sun defines Java as: "a simple, object-oriented, distributed, interpreted, robust, secure, architecture-neutral, portable, high-performance, multithreaded, dynamic, buzzword-compliant, general-purpose programming language. Java supports programming for the Internet in the form of platform-

independent Java applets." Java applets are small, specialized applications that comply with Sun's Java Application Programming Interface (API) allowing developers to add "interactive content" to Web documents (e.g. simple animations, page adornments, basic games, etc.). Applets execute within a Java-compatible browser (e.g. Netscape Navigator) by copying code from the server to client. From a language standpoint, Java's core feature set is based on C++. Sun's Java literature states that Java is basically "C++, with extensions from Objective C for more dynamic method resolution".

Another technology that provides similar function to JAVA is provided by Microsoft and ActiveX Technologies, to give developers and Web designers wherewithal to build dynamic content for the Internet and personal computers. ActiveX includes tools for developing animation, 3-D virtual reality, video and other multimedia content. The tools use Internet standards, work on multiple platforms, and are being supported by over 100 companies. The group's building blocks are called ActiveX Controls, small, fast components that enable developers to embed parts of software in hypertext markup language (HTML) pages. ActiveX Controls work with a variety of programming languages including Microsoft Visual C++, Borland Delphi, Microsoft Visual Basic programming system and, in the future, Microsoft's development tool for Java, code named "Jakarta." ActiveX Technologies also includes ActiveX Server Framework, allowing developers to create server applications. One of ordinary skill in the art will readily recognize that ActiveX could be substituted for JAVA without undue experimentation to practice the invention.

Figure **1B** depicts an overview of the present invention. Customer computer system **120** is in communication with merchant computer

system **130**. The customer-merchant session **150** operates under a general-purpose secure communication protocol such as the SSL protocol. Merchant computer system **130** is additionally in communication with payment gateway computer system **140**. A payment gateway is a system that provides electronic commerce services in support of a bank or other financial institution, and that interfaces to the financial institution to support the authorization and capture of transactions. The customer-institution session **170** operates under a variant of a secure payment technology such as the SET protocol, as described herein, referred to as Merchant-Originated Secure Electronic Transactions ("MOSET"), as is more fully described herein.

Customer-to-Merchant Communication

Figure **2** depicts a more detailed view of customer computer system **120** in communication with merchant system **130** using customer-merchant session **150** operating under the SSL protocol as documented in Freier and incorporated by reference.

Customer computer system **120** initiates communication with merchant computer system **130** using any well-known access protocol, e.g., Transmission Control Protocol/Internet Protocol ("TCP/IP"). In this implementation, customer computer system **120** acts as a client and merchant computer system **130** acts as a server.

Customer computer system **120** initiates communication by sending "client hello" message **210** to the merchant computer system **130**. When a client first connects to a server it is required to send the client hello message **210** as its first message. The client can also send a client hello message **210** in response to a hello request on its own

initiative in order to renegotiate the security parameters in an existing connection. The client hello message includes a random structure, which is used later in the protocol. Specifically, the random structure includes the current time and date in standard UNIX 32-bit format according to the sender's internal clock and twenty-eight bytes of data generated by a secure random number generator. The client hello message **210** further includes a variable length session identifier. If not empty, the session identifier value identifies a session between the same client and server whose security parameters the client wishes to reuse. The session identifier may be from an earlier connection, the current connection, or another currently active connection. It is useful to specify the current connection if the client only wishes to update the random structures and derived values of a connection. It is useful to specify another currently active connection if the client wishes to establish several simultaneous independent secure connections to the same server without repeating the full handshake protocol. Client hello message **210** further includes an indicator of the cryptographic algorithms supported by the client in order of the client's preference, ordered according to client preference.

In response to client hello message **210**, if merchant computer system **130** wishes to correspond with customer computer system **120**, it responds with server hello message **215**. If merchant computer system **130** does not wish to communicate with customer computer system **120**, it responds with a message, not shown, indicating refusal to communicate.

Server hello message **215** includes a random structure, which is used later in the protocol. The random structure in server hello message **215** is in the same format as, but has contents independent of, the

random structure in client hello message **210**. Specifically, the random structure includes the current time and date in standard UNIX 32-bit format according to the sender's internal clock and twenty-eight bytes of data generated by a secure random number generator. Server hello message **215** further includes a variable length session identifier. The session identifier value identifies a new or existing session between the same client and server. Server hello message **215** further includes an indicator of the cryptographic algorithms selected from among the algorithms specified by client hello message **210**, which will be used in further encrypted communications.

Optionally, Merchant computer system **130** transmits a server certificate **220**. If transmitted, server certificate **130** enables customer computer system **120** to authenticate the identity of merchant computer system **130**.

If merchant computer system **130** does not transmit a server certificate **220**, or if server certificate **220** is suitable only for authentication, it may optionally transmit a server key exchange message **225**. Server key exchange message **225** identifies a key that may be used by customer computer system **120** to decrypt further messages sent by merchant computer system **130**.

After transmitting server hello message **215**, and optionally transmitting server certificate **220** or server key exchange message **225**, merchant computer system **130** transmits a server hello done message **230** and waits for a further response from customer computer system **120**.

Customer computer system **120** optionally transmits client certificate **240** to merchant computer system **130**. If transmitted, client certificate **240** enables merchant computer system **130** to authenticate the identity of customer computer system **120**. Alternatively, customer computer system **120** may transmit a no-client-certificate alert **245**, to indicate that the customer has not registered with any certification authority.

If customer computer system **130** does not transmit a client certificate **240**, or if client certificate **240** is suitable only for authentication, customer computer system **130** may optionally transmit a client key exchange message **250**. Client key exchange message **250** identifies a key that may be used by merchant computer system **130** to decrypt further messages sent by customer computer system **120**.

After optionally transmitting client certificate **240**, no-client-certificate alert **245**, and/or client key exchange message **250**, customer computer system **120** transmits a finished message **260**.

At this point, customer computer system **120** and merchant computer system **130** have:

1) negotiated an encryption scheme that may be commonly employed in further communications, and

2) have communicated to each other a set of encryption keys that may be used to decrypt further communications between the two computer systems.

Customer computer system **120** and merchant computer system **130** may thereafter engage in secure communications **270** with less risk of interception by third parties.

Among the messages communicated by customer computer system **120** to merchant computer system **130** may be messages that specify goods or services to be ordered and payment information, such as a credit card number and related information, collectively referred to as "payment information," that may be used to pay for the goods and/or services ordered. In order to obtain payment, the merchant must supply this information to the bank or other payment gateway responsible for the proffered payment method. This enables the merchant to perform payment authorization and payment capture. Payment authorization is the process by which permission is granted by a payment gateway operating on behalf of a financial institution to authorize payment on behalf of the financial institution. This is a process that assesses transaction risk, confirms that a given transaction does not raise the account holder's debt above the account's credit limit, and reserves the specified amount of credit. Payment capture is the process that triggers the movement of funds from the financial institution to the merchant's account.

Payment Authorization

Figure **3** depicts an overview of the method of securely supplying payment information to a payment gateway in order to obtain payment authorization. In function block **310**, merchant computer system **130** generates a payment authorization request **315** and transmits it to payment gateway computer system **140**. In function block **330**, payment gateway system **140** processes the payment authorization request, generates a payment authorization response **325** and transmits it to merchant computer system **130**. In function block **320**, merchant computer system **130** processes payment authorization response **325** and determines whether payment for the

goods or services sought to be obtained by the customer has been authorized.

Payment Authorization Request Generation

Figure **4** depicts the detailed steps of generating and transmitting a payment authorization request. Figures **5A** through **5F** depict views of the payment authorization request and its component parts. In function block **410**, merchant computer system **130** creates a basic authorization request **510**. The basic authorization request is a data area that includes all the information for determining whether a request should be granted or denied. Specifically, it includes such information as the party who is being charged, the amount to be charged, the account number of the account to be charged, and any additional data, such as passwords, needed to validate the charge. This information is either calculated based upon prior customer merchandise selection, or provided by the customer over the secure link **270** established in the customer-merchant general-purpose secure communication protocol session. Fig **5A** depicts a basic authorization request **510**.

In function block **420**, merchant computer system **130** combines basic authorization request **510**, a copy of its encryption public key certificate **515** and a copy of its signature public key certificate **520**. Merchant computer system **130** calculates a digital signature **525** for the combined contents of the combined block **530** comprising basic authorization request **510**, the encryption public key certificate **515** and the signature public key certificate **520**, and appends it to the combination of the combined basic authorization request **510**, the encryption public key certificate **515** and the signature public key certificate **520**. The merchant computer system calculates digital

signature **525** by first calculating a "message digest" based upon the contents of the combined basic authorization request **510**, the encryption public key certificate **515** and the signature public key certificate **520**. A message digest is the fixed-length result that is generated when a variable length message is fed into a one-way hashing function. Message digests help verify that a message has not been altered because altering the message would change the digest. The message digest is then encrypted using the merchant computer system's **130** digital signature private key, thus forming a digital signature.

Figure **5B** depicts the combined block **530** formed by function block **420** and containing basic authorization request **510**, the encryption public key certificate **515**, the signature public key certificate **520**, and digital signature **525**.

In function block **430**, merchant computer system **130** generates a random encryption key RK-0 **540**, denoted as RK-0. Random encryption key RK-0 **540** is a symmetric encryption key. A symmetric encryption key is a key characterized by the property that a message encrypted with a symmetric key can be decrypted with that same key. This is contrasted with an asymmetric key pair, such as a public-key/private-key key pair, where a message encrypted with one key of the key pair may only be decrypted with the other key of the same key pair. Figure **5C** depicts random encryption key RK-0 **540**.

In function block **440**, merchant computer system **130** encrypts combined block **530** using random encryption key RK-0 **540** to form encrypted combined block **550**. Figure **5D** depicts encrypted combined block **550**. The encryption state of encrypted combined

block **550** is graphically shown by random key lock **555**, which indicates that encrypted combined block **550** is encrypted using random key RK-0 **540**.

In function block **450**, merchant computer system **130** encrypts random encryption key RK-0 **540** using the public key of payment gateway system **140** to form encrypted random key **560**. Figure **5E** depicts encrypted random key **560**. The encryption state of encrypted random key **560** is graphically shown by payment gateway public key lock **565**, which indicates that encrypted random key **560** is encrypted using the payment gateway public key.

In function block **460**, merchant computer system **130** concatenates encrypted combined block **550** and encrypted random key 560 to form merchant authorization request **315**. Figure **5F** depicts merchant authorization request **315** comprising encrypted combined block **550** and encrypted random key **560**. In function block **470**, merchant computer system **130** transmits merchant authorization request **315** to payment gateway system **140**.

Payment Authorization Request Processing

Figure **6** depicts the detailed steps of processing a payment authorization request and generating and transmitting a payment authorization request response. Function blocks **610** through **630** depict the steps of processing a payment authorization request, while function blocks **635** through **685** depict the steps of generating and transmitting a payment authorization request response.

In function block **610**, payment gateway computer system **140** applies its private key to encrypted random key **560** contained within received merchant authorization request **315**, thereby decrypting it and obtaining a cleartext version of random key RK-0 **540**. In function block **615**, payment gateway computer system **140** applies random key RK-0 **540** to encrypted combined block **550**, thereby decrypting it and obtaining a cleartext version of combined block **530**. It will be recalled that combined block **530** comprises basic authorization request **510**, a copy of merchant computer system's **130** encryption public key certificate **515** and a copy of merchant computer system's **130** signature public key certificate **520**, as well as merchant digital signature **525**.

In function block **620**, payment gateway computer system **140** verifies merchant computer system's **130** encryption public key certificate **515** and merchant computer system's **130** signature public key certificate **520**. Payment gateway computer system **140** performs this verification by making a call to the certification authorities associated with each certificate. If verification of either certificate fails, payment gateway computer system **140** rejects the authorization request.

In function block **625**, payment gateway computer system **140** validates merchant digital signature **525**. Payment gateway computer system **140** performs this validation by calculating a message digest over the contents of the combined basic authorization request **510**, the encryption public key certificate **515** and the signature public key certificate **520**. Payment gateway computer system **140** then decrypts digital signature **525** to obtain a copy of the equivalent message digest calculated by merchant computer system **130** in function block **420**. If the two message digests are equal, the digital signature **525** is

validated. If validation fails, payment gateway computer system **140** rejects the authorization request.

In function block **630**, payment gateway computer system **140** determines the financial institution for which authorization is required by inspection of basic authorization request **510**. Payment gateway computer system **140** contacts the appropriate financial institution using a secure means, e.g, a direct-dial modem-to-modem connection, or a proprietary internal network that is not accessible to third parties, and using prior art means, obtains a response indicating whether the requested payment is authorized.

Payment Authorization Response Generation

Function blocks **635** through **685** depict the steps of generating and transmitting a payment authorization request response. Figures **7A** through **7J** depict views of the payment authorization response and its component parts.

In function block **635**, payment gateway computer system **140** creates a basic authorization response **710**. The basic authorization request is a data area that includes all the information to determine whether a request was granted or denied. Figure **7A** depicts basic authorization response **710**.

In function block **640**, payment gateway computer system **140** combines basic authorization response **710**, and a copy of its signature public key certificate **720**. Payment computer system **140** calculates a digital signature **725** for the combined contents of the combined block **730** comprising basic authorization response **710** and the signature public key certificate **720**, and appends the signature to

the combination of the combined basic authorization response **710** and the signature public key certificate **720**. The payment gateway computer system calculates digital signature **725** by first calculating a message digest based on the contents of the combined basic authorization response **710** and signature public key certificate **720**. The message digest is then encrypted using the merchant computer system's **140** digital signature private key, thus forming a digital signature.

Figure **7B** depicts the combined block **730** formed in function block **640** and containing basic authorization response **710**, the signature public key certificate **720**, and digital signature **725**.

In function block **645**, payment gateway computer system **150** generates a first symmetric random encryption key **740**, denoted as RK-1. Figure **7C** depicts first random encryption key RK-1 **740**.

In function block **650**, payment gateway computer system **140** encrypts combined block **730** using random encryption key RK-1 **740** to form encrypted combined block **750**. Figure **7D** depicts encrypted combined block **750**. The encryption state of encrypted combined block **750** is graphically shown by random key lock **755**, which indicates that encrypted combined block **750** is encrypted using random key RK-1 **740**.

In function block **655**, payment gateway computer system **140** encrypts random encryption key RK-1 **740** using the public key of merchant computer system **130** to form encrypted random key RK **760**. Figure **7E** depicts encrypted random key RK-1 **760**. The encryption state of encrypted random key **760** is graphically shown by

merchant public key lock **765**, which indicates that encrypted random key **760** is encrypted using the merchant public key.

In function block **660**, payment gateway computer system **140** generates a random capture token **770**. Random capture token **770** will be used in subsequent payment capture processing to associate the payment capture request with the payment authorization request being processed. Figure **7F** depicts capture token **775**.

In function block **665**, payment gateway computer system **140** generates a second symmetric random encryption key **775**, denoted as RK-2. Figure **7G** depicts second random encryption key RK-2 **775**.

In function block **670**, payment gateway computer system **140** encrypts capture token **770** using random encryption key RK-2 **770** to form encrypted capture token **780**. Figure **7H** depicts encrypted capture token **780**. The encryption state of encrypted capture token **780** is graphically shown by random key lock **785**, which indicates that encrypted capture token **780** is encrypted using random key RK-2 **770**.

In function block **675**, payment gateway computer system **140** encrypts second random encryption key RK-2 **775** using its own public key to form encrypted random key RK-2 **790**. Figure **7I** depicts encrypted random key RK-2 **790**. The encryption state of encrypted random key **790** is graphically shown by payment gateway public key lock **795**, which indicates that encrypted random key **790** is encrypted using the payment gateway public key.

In function block **680**, payment gateway computer system **140** concatenates encrypted combined block **750**, encrypted random key RK-1 **760**, encrypted capture token **780** and encrypted random key RK-2 **790** to form merchant authorization response **325**. Figure **7J** depicts merchant authorization response **325** comprising encrypted combined block **750**, encrypted random key RK-1 **760**, encrypted capture token **780** and encrypted random key RK-2 **790**. In function block **685**, payment gateway computer system **140** transmits merchant authorization response **325** to merchant system **130**.

Payment Authorization Response Processing

Figure **8** depicts the detailed steps of processing a payment authorization response. In function block **810**, merchant computer system **130** applies its private key to encrypted random key RK-1 **760** contained within received merchant authorization response **325**, thereby decrypting it and obtaining a cleartext version of random key RK-1 **740**.

In function block **820**, merchant computer system **130** applies random key RK-1 **740** to encrypted combined block **750**, thereby decrypting it and obtaining a cleartext version of combined block **730**. It will be recalled that combined block **730** comprises basic authorization response **710**, a copy of payment gateway computer system's **140** signature public key certificate **720**, as well as payment gateway digital signature **725**.

In function block **830**, merchant computer system **130** verifies payment gateway computer system's **140** signature public key certificate **720**. Merchant computer system **130** performs this verification by making a call to the certification authority associated

with the certificate. If verification of the certificate fails, merchant computer system **130** concludes that the authorization response is counterfeit and treats it though the authorization request had been rejected.

In function block **840**, merchant computer system **130** validates payment gateway digital signature **725**. Merchant computer system **130** performs this validation by calculating a message digest over the contents of the combined basic authorization request **710** and the signature public key certificate **720**. Merchant computer system **130** then decrypts digital signature **725** to obtain a copy of the equivalent message digest calculated by payment gateway computer system **140** in function block **640**. If the two message digests are equal, the digital signature **725** is validated. If validation fails, concludes that the authorization response is counterfeit and treats it though the authorization request had been rejected.

In function block **850**, merchant computer system **130** stores encrypted capture token **780** and encrypted random key RK-2 **790** for later use in payment capture. In function block **860**, merchant computer system **130** processes the customer purchase request in accordance with the authorization response **710**. If the authorization response indicates that payment in authorized, merchant computer system **130** fills the requested order. If the authorization response indicates that payment is not authorized, or if merchant computer system **130** determined in function block **830** or **840** that the authorization response is counterfeit , merchant computer system **130** indicates to the customer that the order cannot be filled.

Payment Capture

Figure **9** depicts an overview of the method of securely supplying payment capture information to payment gateway **140** in order to obtain payment capture. In function block **910**, merchant computer system **130** generates a merchant payment capture request **915** and transmits it to payment gateway computer system **140**. In function block **930**, payment gateway system **140** processes the payment capture request **915**, generates a payment capture response **925** and transmits it to merchant computer system **130**. In function block **920**, merchant computer system **130** processes payment capture response **925** and verifies that payment for the goods or services sought to be obtained by the customer have been captured.

Payment Capture Request Generation

Figure **10** depicts the detailed steps of generating and transmitting a payment capture request. Figures **11A** through **11F** depict views of the payment capture request and its component parts. In function block **1010**, merchant computer system **130** creates a basic capture request **510**. The basic capture request is a data area that includes all the information needed by payment gateway computer system **140** to trigger a transfer of funds to the merchant operating merchant computer system **130**.

Specifically, a capture request includes a capture request amount, a capture token, a date, summary information of the purchased items and a Merchant ID (MID) for the particular merchant. Figure **11A** depicts basic authorization request **1110**.

In function block **1020**, merchant computer system **130** combines basic capture request **1110**, a copy of its encryption public key certificate **1115** and a copy of its signature public key certificate

1120. Merchant computer system **130** calculates a digital signature **1125** for the combined contents of the combined block **1130** comprising basic capture request **1110**, the encryption public key certificate **1115** and the signature public key certificate **1120**, and appends it to the combination of the combined basic capture request **1110**, the encryption public key certificate **1115** and the signature public key certificate **1120**. The merchant computer system calculates digital signature **1125** by first calculating a message digest over the contents of the combined basic capture request **1110**, the encryption public key certificate **1115** and the signature public key certificate **1120**. The message digest is then encrypted using the merchant computer system's **130** digital signature private key, thus forming a digital signature.

Figure **11B** depicts the combined block **1130** formed by function block **1020** and containing basic capture request **1110**, the encryption public key certificate **1115**, the signature public key certificate **1120**, and digital signature **1125**.

In function block **1030**, merchant computer system **130** generates a random encryption key **1140**, denoted as RK-3. Random encryption key RK-3 **1140** is a symmetric encryption key. Figure **11C** depicts random encryption key RK-3 **1140**.

In function block **1040**, merchant computer system **130** encrypts combined block **1130** using random encryption key RK-3 **1140** to form encrypted combined block **1150**. Figure **11D** depicts encrypted combined block **1150**. The encryption state of encrypted combined block **1150** is graphically shown by random key lock **1155**, which

indicates that encrypted combined block **1150** is encrypted using random key RK-3 **1140**.

In function block **1050**, merchant computer system **130** encrypts random encryption key RK-3 **1140** using the public key of payment gateway system **140** to form encrypted random key **1160**. Figure **11E** depicts encrypted random key **1160**. The encryption state of encrypted random key **1160** is graphically shown by payment gateway public key lock **1165**, which indicates that encrypted random key RK-3 **1160** is encrypted using the payment gateway public key.

In function block **1060**, merchant computer system **130** concatenates encrypted combined block **1150**, encrypted random key **1160**, and the encrypted capture token **780** and encrypted random key RK-2 **790** that were stored in function block **850** to form merchant capture request **915**. Figure **11F** depicts merchant capture request **915**, comprising encrypted combined block **1150**, encrypted random key **1160**, encrypted capture token **780** and encrypted random key RK-2 **790**. In function block **1070**, merchant computer system **130** transmits merchant capture request **915** to payment gateway system **140**.

Payment Capture Request Processing

Figure **12** depicts the detailed steps of processing a payment capture request and generating and transmitting a payment capture request response. Function blocks **1210** through **1245** depict the steps of processing a payment capture request, while function blocks **1250** through **1285** depict the steps of generating and transmitting a payment capture request response.

In function block **1210**, payment gateway computer system **140** applies its private key to encrypted random key **1160** contained within received merchant capture request **915**, thereby decrypting it and obtaining a cleartext version of random key RK-3 **1140**. In function block **1215**, payment gateway computer system **140** applies random key RK-3 **1140** to encrypted combined block **1150**, thereby decrypting it and obtaining a cleartext version of combined block **1130**. It will be recalled that combined block **1130** comprises basic capture request **1110**, a copy of merchant computer system's **130** encryption public key certificate **1115** and a copy of merchant computer system's **130** signature public key certificate **1120**, as well as merchant digital signature **1125**.

In function block **1220**, payment gateway computer system **140** verifies merchant computer system's **130** encryption public key certificate **1115** and merchant computer system's **130** signature public key certificate **1120**. Payment gateway computer system **140** performs this verification by making a call to the certification authorities associated with each certificate. If verification of either certificate fails, payment gateway computer system **140** rejects the capture request.

In function block **1225**, payment gateway computer system **140** validates merchant digital signature **1125**. Payment gateway computer system **140** performs this validation by calculating a message digest over the contents of the combined basic capture request **1110**, the encryption public key certificate **1115** and the signature public key certificate **1120**. Payment gateway computer system **140** then decrypts digital signature **1125** to obtain a copy of the equivalent message digest calculated by merchant computer

system **130** in function block **1020**. If the two message digests are equal, the digital signature **1125** is validated. If validation fails, payment gateway computer system **140** rejects the capture request.

In function block **1230**, payment gateway computer system **140** applies its private key to encrypted random key RK-2 **790** contained within received merchant capture request **915**, thereby decrypting it and obtaining a cleartext version of random key RK-2 **775**. In function block **1235**, payment gateway computer system **140** applies random key RK-2 **775** to encrypted capture token **780**, thereby decrypting it and obtaining a cleartext version of capture token **770**.

In function block **1240**, payment gateway computer system **140** verifies that a proper transaction is being transmitted between capture token **780** and capture request **1110**. A capture token contains data that the gateway generates at the time of authorization. When the authorization is approved, the encrypted capture token is given to the merchant for storage. At the time of capture, the merchant returns the capture token to the gateway along with other information required for capture. Upon receipt of the capture token, the gateway compares a message made of the capture request data and the capture token data and transmits this information over a traditional credit/debit network. If an improperly formatted transaction is detected, payment gateway computer system **140** rejects the capture request.

In function block **1245**, payment gateway computer system **140** determines the financial institution for which capture is requested by inspection of basic capture request **1110**. Payment gateway computer system **140** contacts the appropriate financial institution using a

secure means, e.g, a direct-dial modem-to-modem connection, or a proprietary internal network that is not accessible to third parties, and using prior art means, instructs a computer at the financial institution to perform the requested funds transfer.

Payment Capture Response Generation

Function blocks **1250** through **1285** depict the steps of generating and transmitting a payment capture request response. Figures **13A** through **13F** depict views of the payment capture response and its component parts.

In function block **1250**, payment gateway computer system **140** creates a basic capture response **710**. The basic capture request is a data area that includes all the information to indicate whether a capture request was granted or denied. Figure **13A** depicts basic authorization request **1310**.

In function block **1255**, payment gateway computer system **140** combines basic capture response **1310**, and a copy of its signature public key certificate **1320**. Payment computer system **140** calculates a digital signature **1325** for the combined contents of the combined block **1330** comprising basic capture response **1310** and the signature public key certificate **1320**, and appends the signature to the combination of the combined basic authorization request **1310** and the signature public key certificate **1320**. The payment gateway computer system calculates digital signature **1325** by first calculating a message digest over the contents of the combined basic capture response **1310** and signature public key certificate **720**. The message digest is then encrypted using the merchant computer system's **140** digital signature private key, thus forming a digital signature.

Figure **13B** depicts the combined block **1330** formed by function block **1255** and containing basic capture request **1310**, the signature public key certificate **1320**, and digital signature **1325**.

In function block **1260**, payment gateway computer system **140** generates a symmetric random encryption key **1340**, denoted as RK-4. Figure **13C** depicts random encryption key RK-4 **1340**.

In function block **1275**, payment gateway computer system **140** encrypts combined block **1330** using random encryption key RK-4 **1340** to form encrypted combined block **1350**. Figure **13D** depicts encrypted combined block **1350**. The encryption state of encrypted combined block **1350** is graphically shown by random key lock **1355**, which indicates that encrypted combined block **1350** is encrypted using random key RK-4 **1340**.

In function block **1275**, payment gateway computer system **140** encrypts random encryption key RK-4 **1340** using the public key of merchant computer system **130** to form encrypted random key RK-4 **1360**. Figure **13E** depicts encrypted random key RK-4 **1360**. The encryption state of encrypted random key **1360** is graphically shown by merchant public key lock **1365**, which indicates that encrypted random key **1360** is encrypted using the merchant public key.

In function block **1280**, payment gateway computer system **140** concatenates encrypted combined block **1350** and encrypted random key RK-4 **1360** to form merchant capture response **925**. Figure **13F** depicts merchant capture response **925** comprising encrypted combined block **1350** and encrypted random key RK-4 **1360**. In

function block **1285**, payment gateway computer system **140** transmits merchant capture response **925** to merchant system **130**.

Payment Capture Response Processing

Figure **14** depicts the detailed steps of processing a payment capture response. In function block **1410**, merchant computer system **130** applies its private key to encrypted random key RK-4 **1360** contained within received merchant capture response **925**, thereby decrypting it and obtaining a cleartext version of random key RK-4 **1340**.

In function block **1420**, merchant computer system **130** applies random key RK-4 **1340** to encrypted combined block **1350**, thereby decrypting it and obtaining a cleartext version of combined block **1330**. It will be recalled that combined block **1330** comprises basic capture response **1310**, a copy of payment gateway computer system's **140** signature public key certificate **1320**, as well as payment gateway digital signature **1325**.

In function block **1430**, merchant computer system **130** verifies payment gateway computer system's **140** signature public key certificate **1320**. Merchant computer system **130** performs this verification by making a call to the certification authority associated with the certificate. If verification of the certificate fails, merchant computer system **130** concludes that the capture response is counterfeit and raises an error condition.

In function block **1440**, merchant computer system **130** validates payment gateway digital signature **1325**. Merchant computer system

130 performs this validation by calculating a message digest over the contents of the combined basic authorization request **1310** and the signature public key certificate **1320**. Merchant computer system **130** then decrypts digital signature **1325** to obtain a copy of the equivalent message digest calculated by payment gateway computer system **140** in function block **1255**. If the two message digests are equal, the digital signature **1325** is validated. If validation fails, merchant computer system **130** concludes that the authorization response is counterfeit and raises an error condition.

In function block **1450**, merchant computer system **130** stores capture response for later use in by legacy system accounting programs, e.g. to perform reconciliation between the merchant operating merchant computer system **130** and the financial institution from whom payment was requested, thereby completing the transaction.

The system of the present invention permits immediate deployment of a secure payment technology architecture such as the SET architecture without first establishing a public-key encryption infrastructure for use by consumers. It thereby permits immediate use of SET-compliant transaction processing without the need for consumers to migrate to SET-compliant application software.

All publications and existing subsystems mentioned in this specification are hereby incorporated by reference to the same extent as if each individual publication or existing subsystem were specifically and individually indicated to be incorporated by reference.

CLAIMS

What is claimed is:

1. A method for initiating secure communication between a first and a second computer (120, 130) connected to a network (34, 35) for receiving and transmitting payment information, comprising the steps of:

(a) establishing a communication between said first and said second computer (120, 130) via said network (35);

(b) identifying an encryption procedure and a decryption procedure (210-270) utilized by said first and said second computer (120, 130);

(c) transmitting encrypted payment information from said first computer to said second computer (120, 130);

(d) receiving said encrypted payment information (270) at said second computer (130) and decrypting the payment information utilizing the decryption procedure (270); and

(e) repackaging said payment information to comply with a third party secure protocol for further payment processing (410-565).

2. The method as recited in claim 1, including the step of utilizing the Internet for transmitting information between said first and said second computer systems.

3. The method as recited in claim 1, including the step of transmitting from the second computer system to a third computer system for authorizing or denying credit in the payment processing.

4. The method as recited in claim 1, wherein the secure third party protocol is a Secure Electronic Transaction protocol.

5. A method for initiating secure communication as recited in claim 1, further comprising the steps of:

(a) obtaining client information for use in said secure communication between a first and a second computer;

(b) establishing a communication between said first and said second computer via said network; and

(c) repackaging said payment information to comply with a third party secure protocol for further payment processing.

6. The method as recited in claim 5, including:

(d) transmitting encrypted payment information from said first computer to said second computer;

(e) receiving said encrypted payment information at said second computer and decrypting the payment information utilizing the decryption procedure; and

(f) performing further payment processing on said decrypted information.

7. The method as recited in claim 5, wherein said client information is obtained via a telephone, fax machine or electronic mail.

8. The method as recited in claim 5, wherein an electronic signature is utilized to authenticate payment processing.

9. The method as recited in claim 7, wherein said client information is obtained via a secure general purpose protocol.

10. The method as recited in claim 5, further comprising reversing previous payment transactions.

11. Apparatus for initiating payment in a computer under the control of software with an attached display (38) and an input device (26, 24) connected to a network (34, 35) for receiving and transmitting network information (150, 170), comprising:

(a) means for establishing a communication between said first and said second computer via said network (10-35, 150);

(b) means for identifying an encryption procedure and a decryption procedure (10-35, 120-130, 210-270) utilized by said first and said second computer (120-130, 210-270);

(c) means for transmitting encrypted payment information from said first computer to said second computer (10-35, 270);

(d) means for receiving said encrypted payment information at said second computer and decrypting the payment information utilizing the decryption procedure(10-35, 270); and

(e) means for repackaging said payment information to comply with a Secure Electronic Transaction protocol for further payment processing (10-35; 410-565).

12. The apparatus as recited in claim 11, including means for utilizing the Internet for transmitting information between said first and said second computer systems.

13. The apparatus as recited in claim 12, including means for transmitting from the second computer system to a third computer system for authorizing or denying credit in the payment processing.

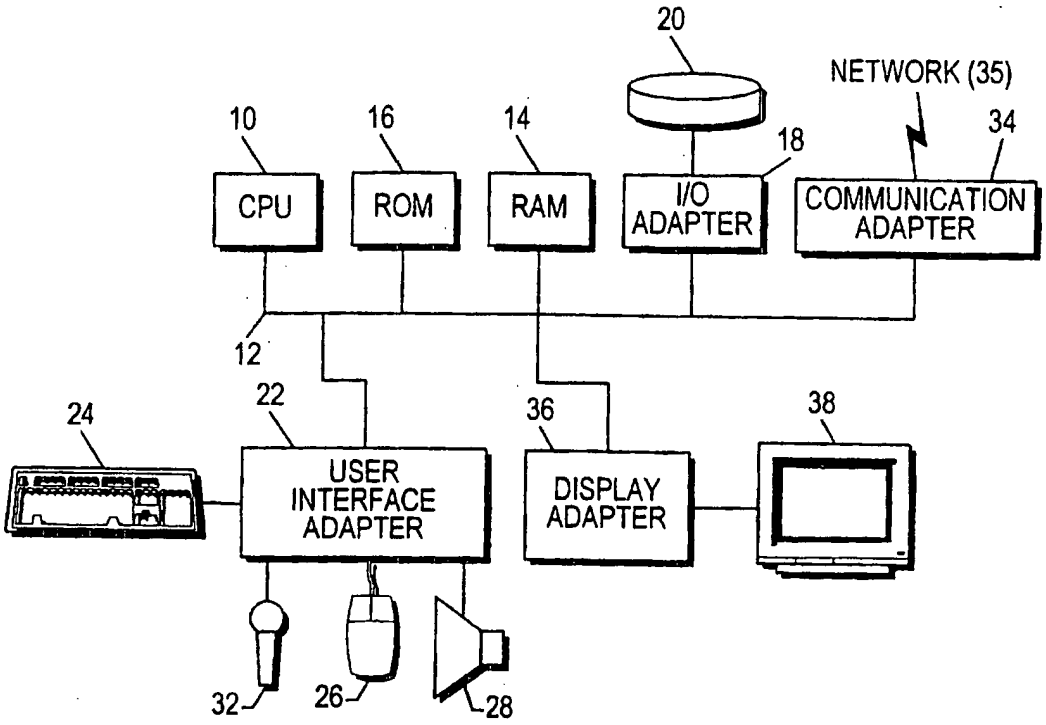

20
NETWORK (35)
10
16
14
18
34
CPU
ROM
RAM
I/O ADAPTER
COMMUNICATION ADAPTER
12
22
36
38
24
USER INTERFACE ADAPTER
DISPLAY ADAPTER
32
26
28

FIG.-1A

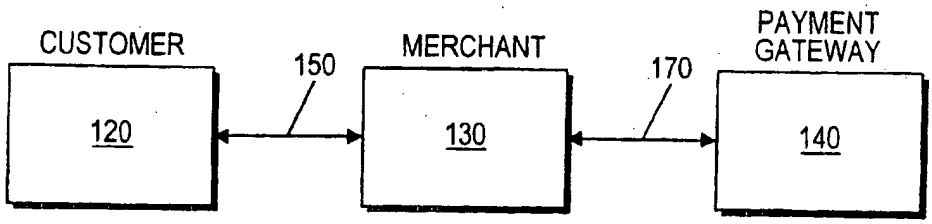

CUSTOMER
150
MERCHANT
170
PAYMENT GATEWAY
120
130
140

FIG.-1B

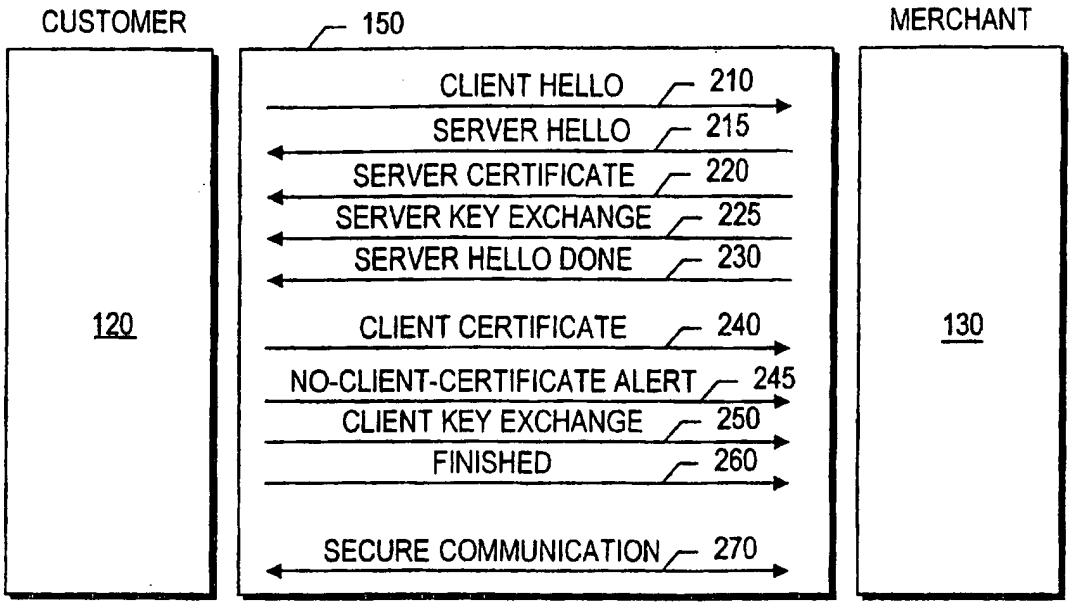

CUSTOMER
150
MERCHANT
CLIENT HELLO 210
SERVER HELLO 215
SERVER CERTIFICATE 220
SERVER KEY EXCHANGE 225
SERVER HELLO DONE 230
120
CLIENT CERTIFICATE 240
NO-CLIENT-CERTIFICATE ALERT 245
CLIENT KEY EXCHANGE 250
FINISHED 260
130
SECURE COMMUNICATION 270

FIG.-2

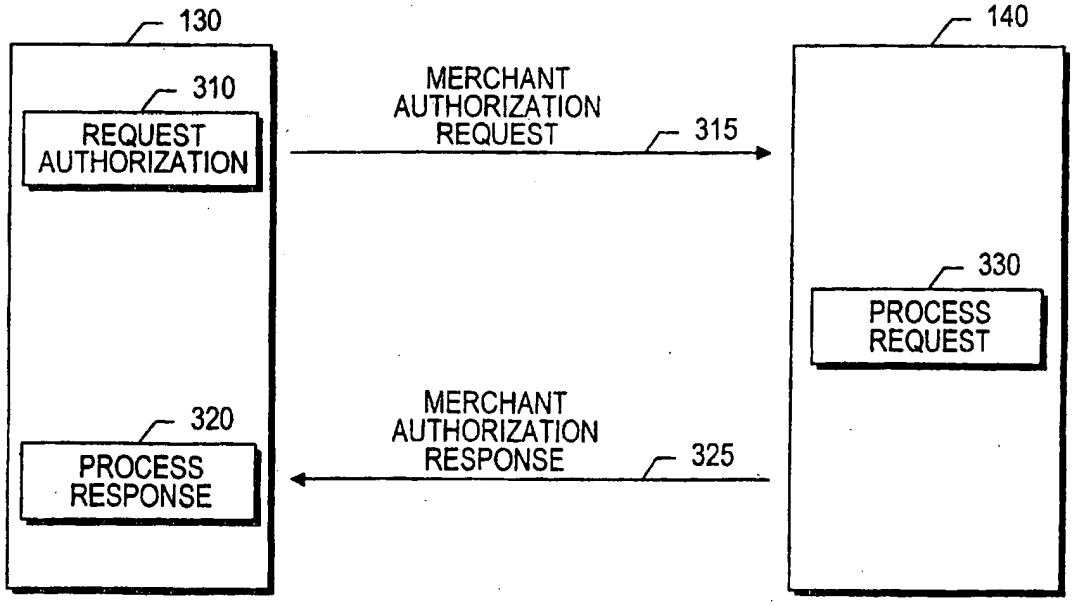

130
140
310
REQUEST AUTHORIZATION
MERCHANT AUTHORIZATION REQUEST 315
330
PROCESS REQUEST
320
PROCESS RESPONSE
MERCHANT AUTHORIZATION RESPONSE 325

FIG.-3

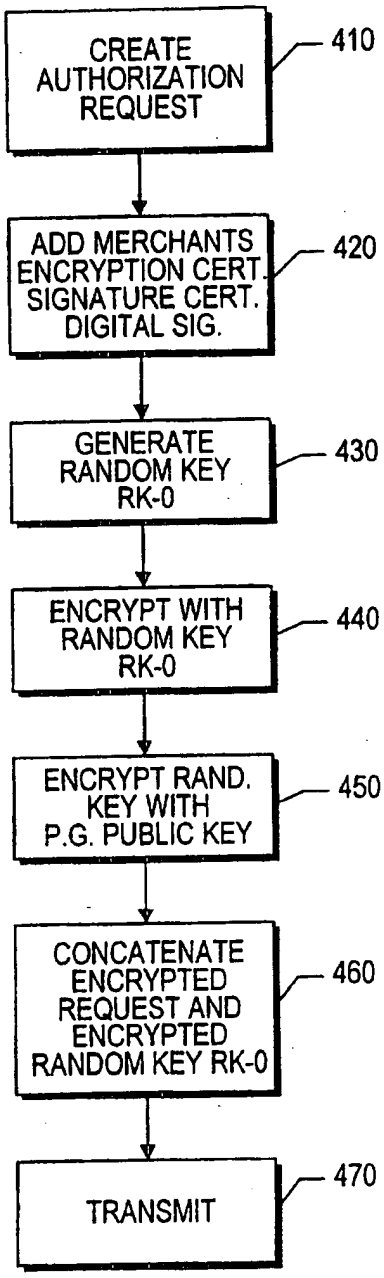

CREATE AUTHORIZATION REQUEST
410
ADD MERCHANTS ENCRYPTION CERT. SIGNATURE CERT. DIGITAL SIG.
420
GENERATE RANDOM KEY RK-0
430
ENCRYPT WITH RANDOM KEY RK-0
440
ENCRYPT RAND. KEY WITH P.G. PUBLIC KEY
450
CONCATENATE ENCRYPTED REQUEST AND ENCRYPTED RANDOM KEY RK-0
460
TRANSMIT
470

FIG.-4

FIG.-5A

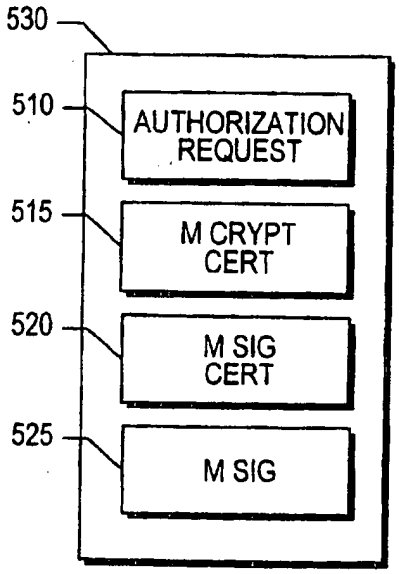


FIG.-5B

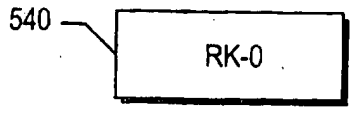


FIG.-5C

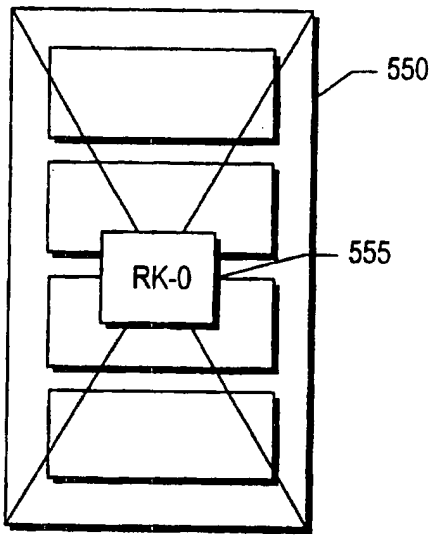


FIG.-5D

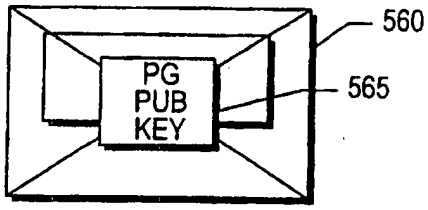


FIG.-5E

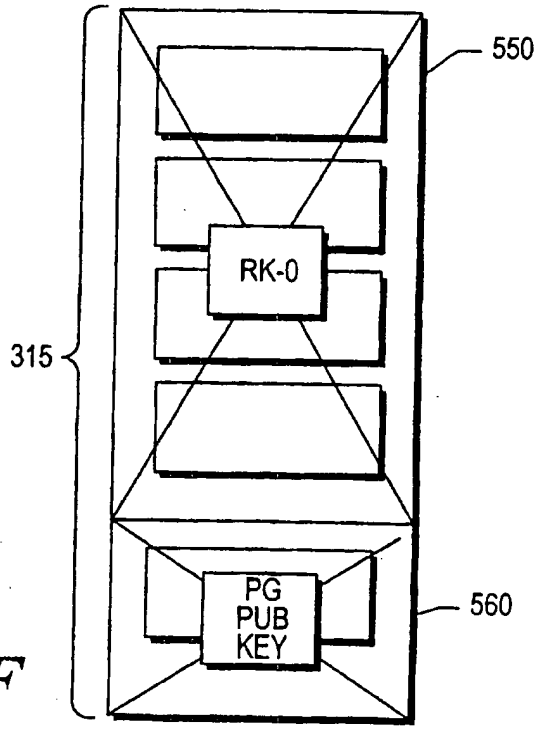


FIG.-5F

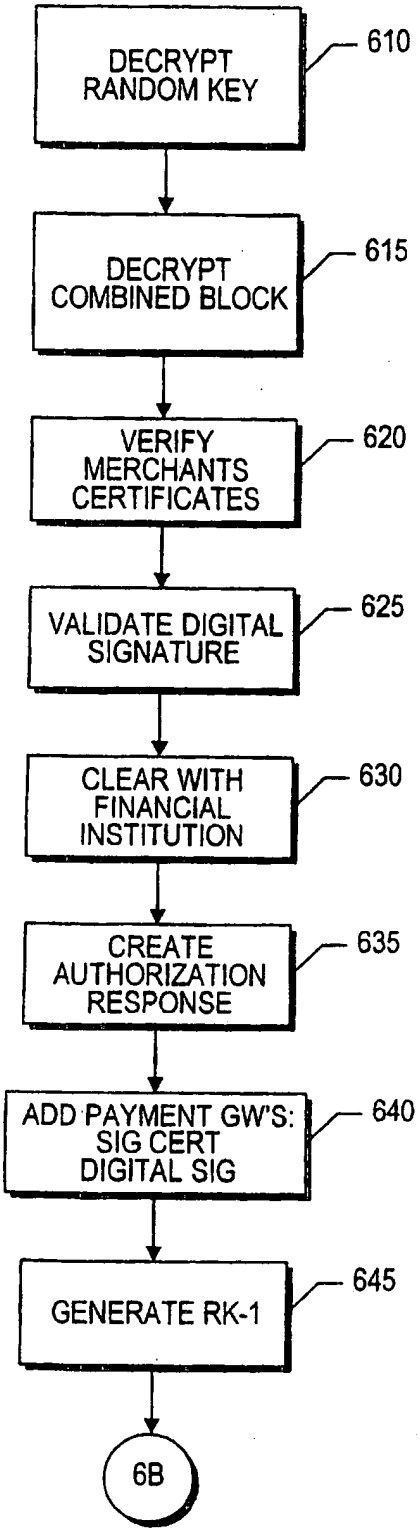

DECRYPT RANDOM KEY
610
DECRYPT COMBINED BLOCK
615
VERIFY MERCHANTS CERTIFICATES
620
VALIDATE DIGITAL SIGNATURE
625
CLEAR WITH FINANCIAL INSTITUTION
630
CREATE AUTHORIZATION RESPONSE
635
ADD PAYMENT GW'S: SIG CERT DIGITAL SIG
640
GENERATE RK-1
645
6B

FIG.-6A

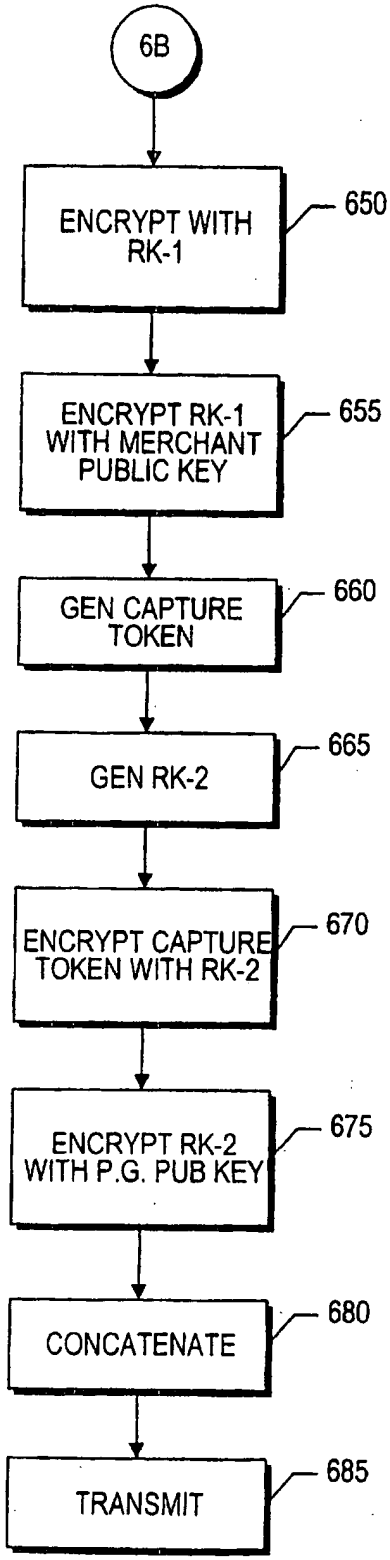

6B
ENCRYPT WITH RK-1
650
ENCRYPT RK-1 WITH MERCHANT PUBLIC KEY
655
GEN CAPTURE TOKEN
660
GEN RK-2
665
ENCRYPT CAPTURE TOKEN WITH RK-2
670
ENCRYPT RK-2 WITH P.G. PUB KEY
675
CONCATENATE
680
TRANSMIT
685

FIG.-6B

FIG.-7A

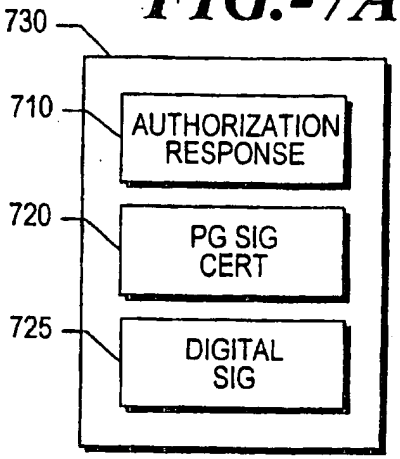


FIG.-7B

FIG.-7C

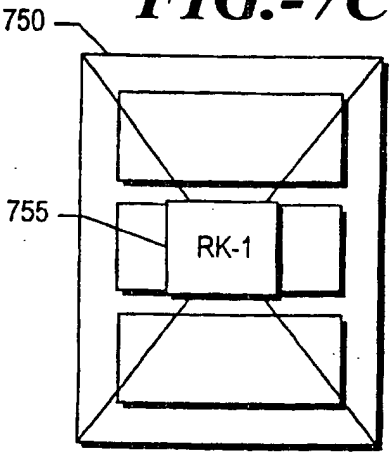


FIG.-7D

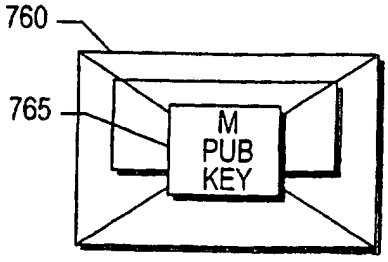


FIG.-7E

FIG.-7F

FIG.-7G

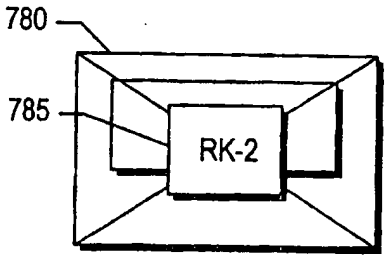


FIG.-7H

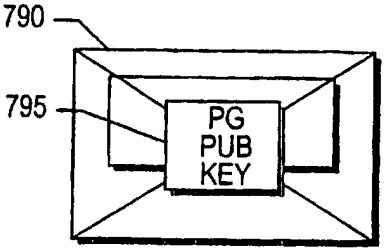


FIG.-7I

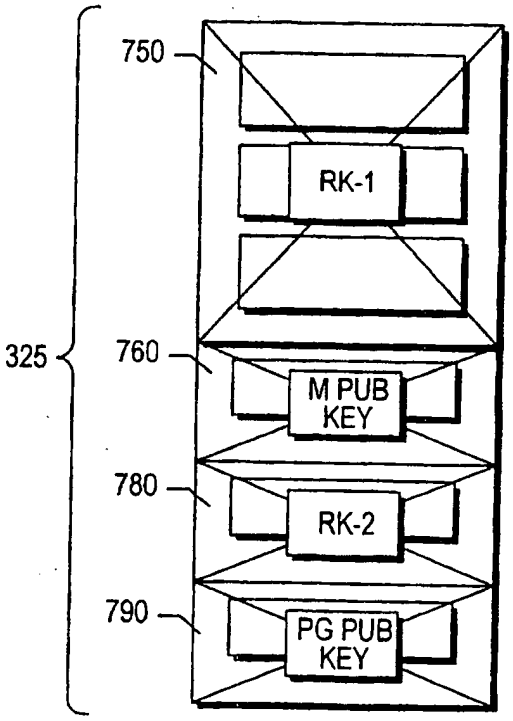


FIG.-7J

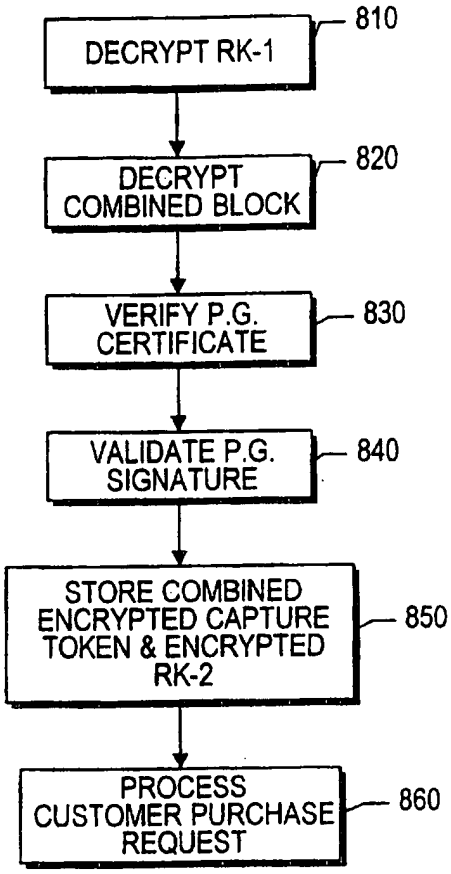

DECRYPT RK-1
810
DECRYPT COMBINED BLOCK
820
VERIFY P.G. CERTIFICATE
830
VALIDATE P.G. SIGNATURE
840
STORE COMBINED ENCRYPTED CAPTURE TOKEN & ENCRYPTED RK-2
850
PROCESS CUSTOMER PURCHASE REQUEST
860

FIG.-8

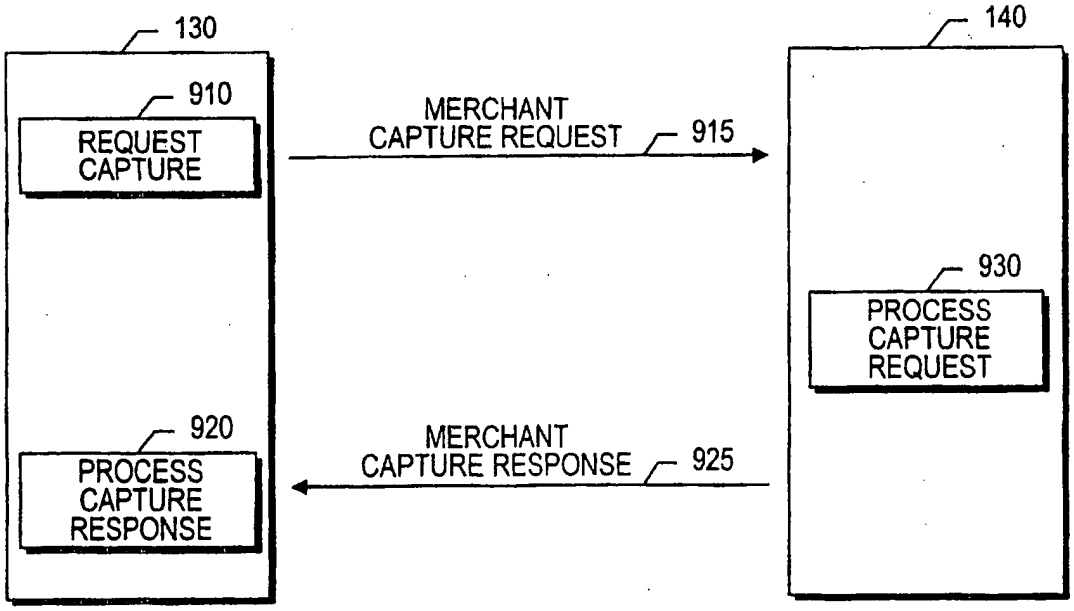

130
910
REQUEST CAPTURE
MERCHANT CAPTURE REQUEST
915
140
930
PROCESS CAPTURE REQUEST
920
PROCESS CAPTURE RESPONSE
MERCHANT CAPTURE RESPONSE
925

FIG.-9

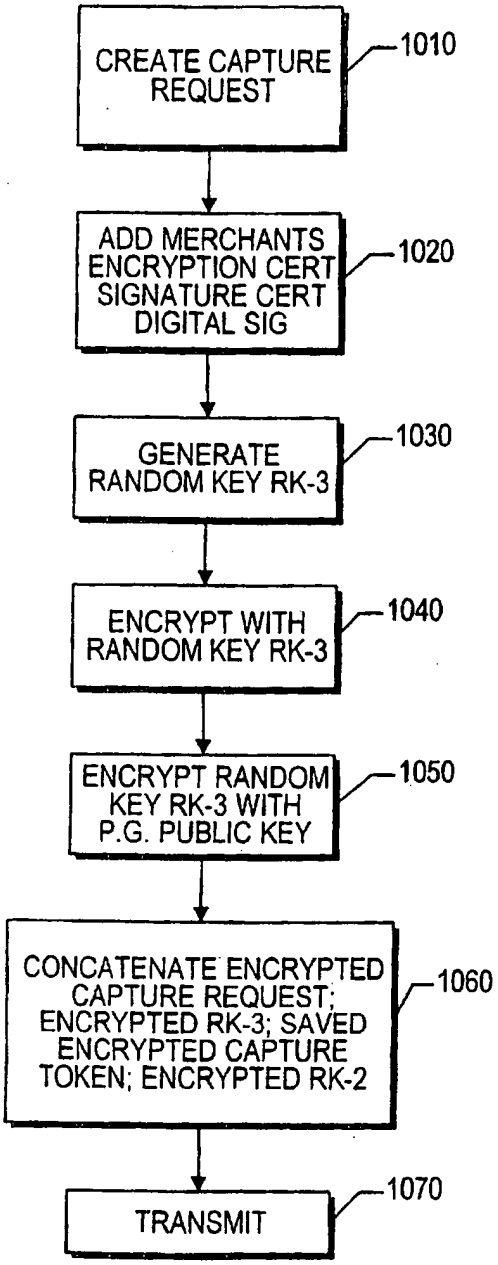

CREATE CAPTURE REQUEST
1010
ADD MERCHANTS ENCRYPTION CERT SIGNATURE CERT DIGITAL SIG
1020
GENERATE RANDOM KEY RK-3
1030
ENCRYPT WITH RANDOM KEY RK-3
1040
ENCRYPT RANDOM KEY RK-3 WITH P.G. PUBLIC KEY
1050
CONCATENATE ENCRYPTED CAPTURE REQUEST; ENCRYPTED RK-3; SAVED ENCRYPTED CAPTURE TOKEN; ENCRYPTED RK-2
1060
TRANSMIT
1070

FIG.-10

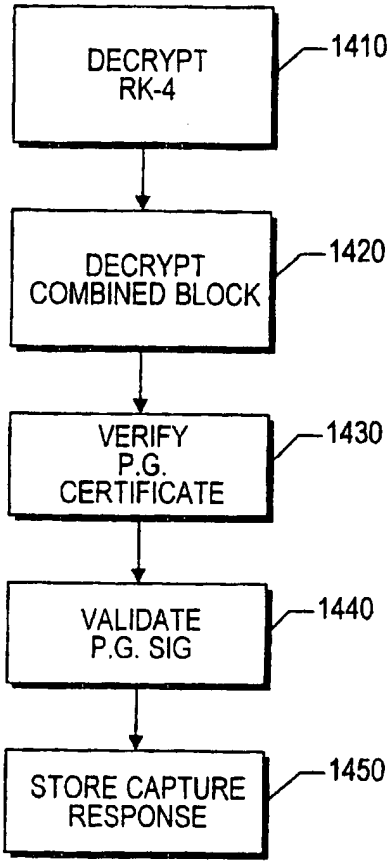

DECRYPT RK-4
1410
DECRYPT COMBINED BLOCK
1420
VERIFY P.G. CERTIFICATE
1430
VALIDATE P.G. SIG
1440
STORE CAPTURE RESPONSE
1450

FIG.-14

FIG.-11A

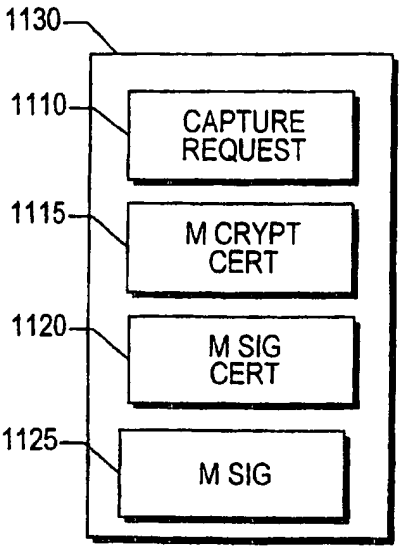


FIG.-11B

FIG.-11C

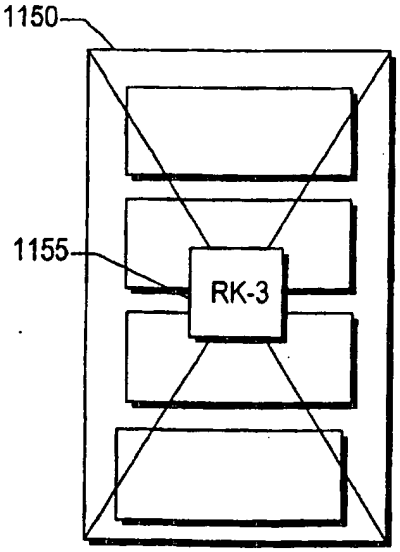


FIG.-11D

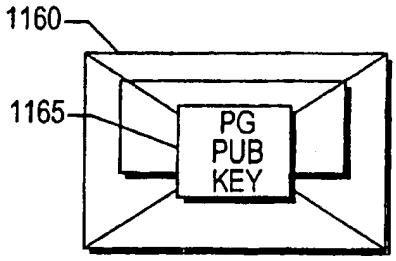


FIG.-11E

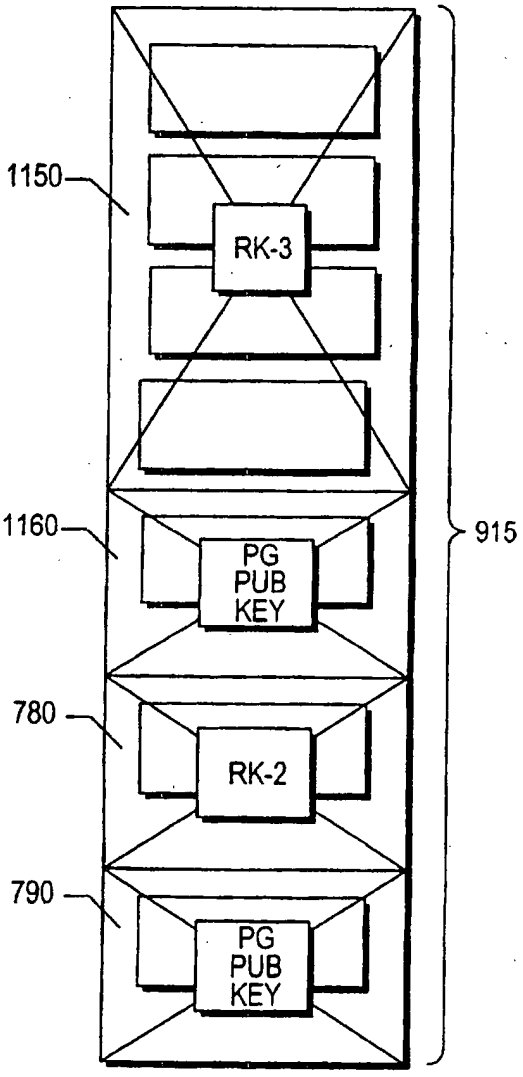


FIG.-11F

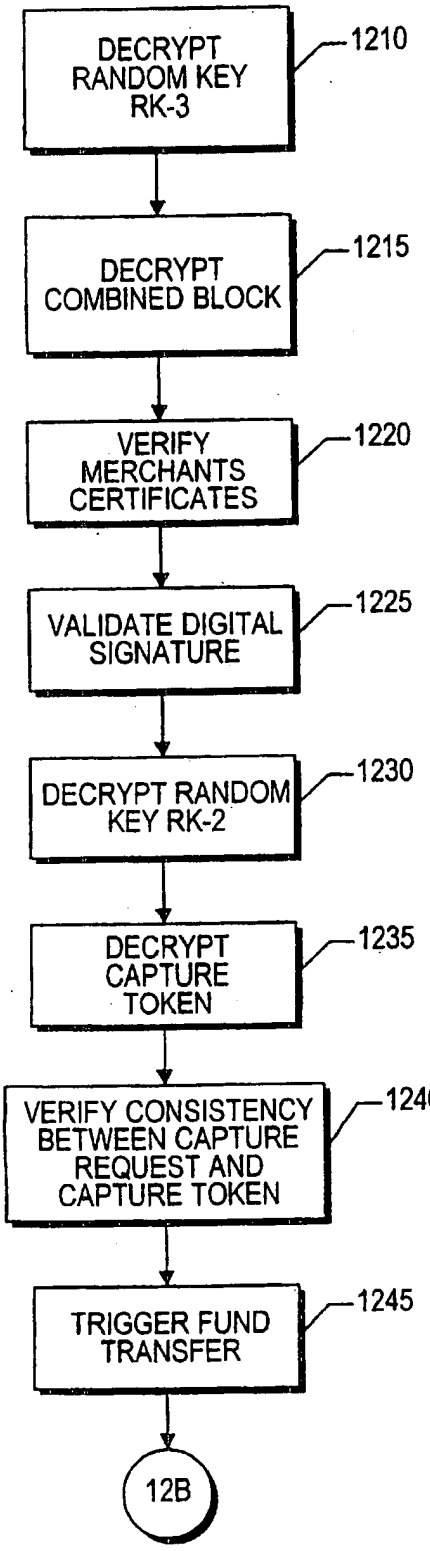


FIG.-12A

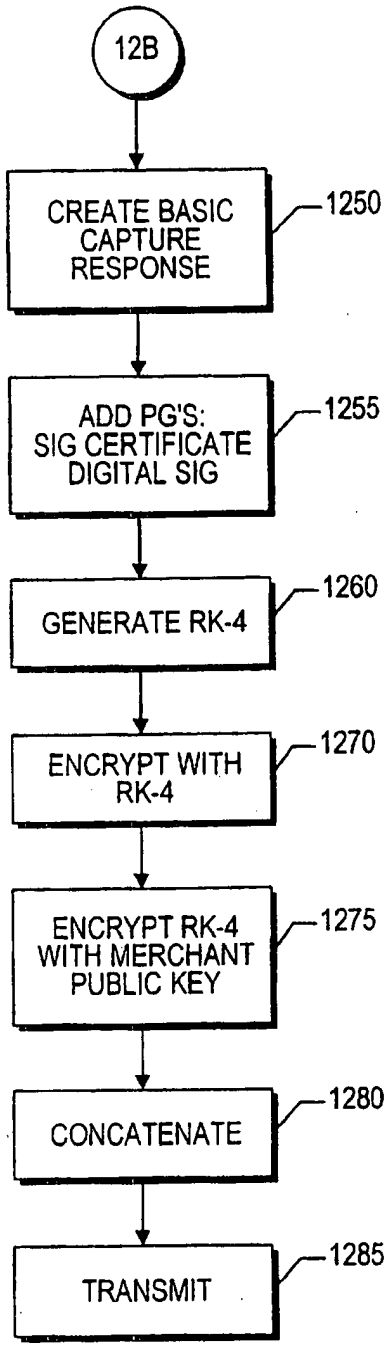


FIG.-12B

FIG.-13A

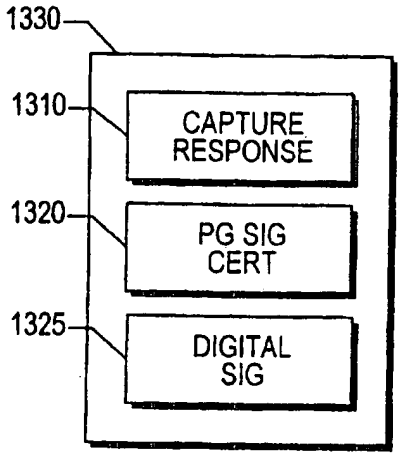


FIG.-13B

FIG.-13C

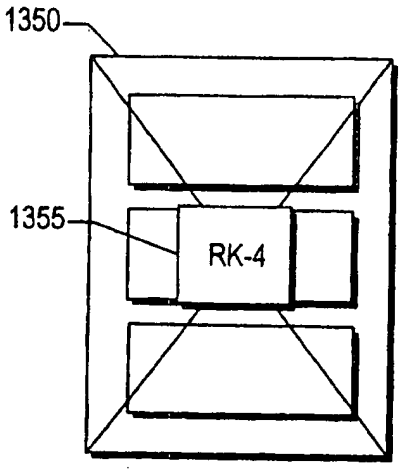


FIG.-13D

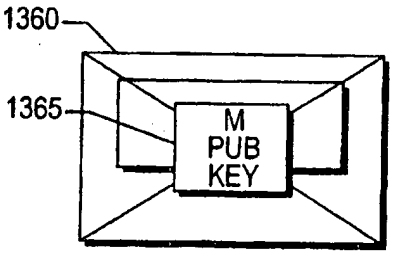


FIG.-13E

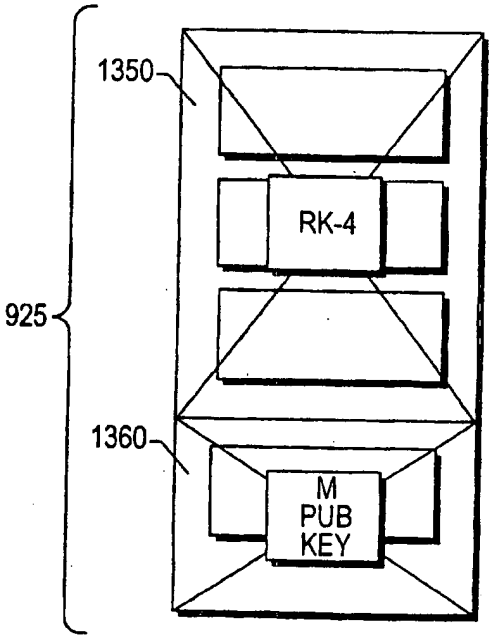


FIG.-13F

INTERNATIONAL SEARCH REPORT

International Application No
PCT/US 97/06934

A. CLASSIFICATION OF SUBJECT MATTER
IPC 6 G07F7/10 H04L9/32

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)
IPC 6 G07F H04L

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practical, search terms used)

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category *	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
	-/--	

[X] Further documents are listed in the continuation of box C.

[ ] Patent family members are listed in annex.

* Special categories of cited documents :

"A" document defining the general state of the art which is not considered to be of particular relevance

"E" earlier document but published on or after the international filing date

"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)

"O" document referring to an oral disclosure, use, exhibition or other means

"P" document published prior to the international filing date but later than the priority date claimed

"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention

"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone

"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art.

"&" document member of the same patent family

Date of the actual completion of the international search	Date of mailing of the international search report
8 September 1997	22.09.97
Name and mailing address of the ISA European Patent Office, P.B. 5818 Patentlaan 2 NL - 2280 HV Rijswijk Tel. (+31-70) 340-2040, Tx. 31 651 epo nl, Fax (+31-70) 340-3016	Authorized officer Holper, G

2

INTERNATIONAL SEARCH REPORT

International Application No
PCT/US 97/06934

C.(Continuation) DOCUMENTS CONSIDERED TO BE RELEVANT

Category *	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	PROCEEDINGS OF THE FIRST USENIX WORKSHOP OF ELECTRONIC COMMERCE, PROCEEDINGS OF THE FIRST USENIX WORKSHOP OF ELECTRONIC COMMERCE, NEW YORK, NY, USA, 11-12 JULY 1995, 1995, BERKELEY, CA, USA, USENIX ASSOC, USA, pages 89-106, XP000579445 BELLARE M ET AL: "iKP-a family of secure electronic payment protocols" see page 90, right-hand column, line 10 - line 17 see page 94, right-hand column, line 4 - line 18 see page 96, left-hand column, line 34 - right-hand column, line 27 see page 99, left-hand column, line 17 - line 37 see page 99, right-hand column, line 1 - line 2 ---	1,2,5,6, 8,11,12
A	BYTE, vol. 20, no. 6, June 1995, PETERBOROUGH (US), pages 71-78, XP000509328 SINGLETON: "CASH ON THE WIREHEAD" see page 72, middle column, last paragraph - right-hand column, line 11 see page 76, right-hand column :DIGICASH ---	1,2,11, 12
A	IEEE PERSONAL COMMUNICATIONS, AUG. 1995, USA, vol. 2, no. 4, ISSN 1070-9916, pages 34-39, XP000577034 SIRBU M ET AL: "NetBill: an Internet commerce system optimized for network-delivered services" see page 21, right-hand column, line 37 - page 22, left-hand column, paragraph 4 see page 23, right-hand column, line 1 - line 9 ---	1,7,11
A	SECURITY PROTOCOLS. INTERNATIONAL WORKSHOP PROCEEDINGS, SECURITY PROTOCOLS. INTERNATIONAL WORKSHOP, CAMBRIDGE, UK, 10-12 APRIL 1996, ISBN 3-540-62494-5, 1997, BERLIN, GERMANY, SPRINGER-VERLAG, GERMANY, pages 49-57, XP002039777 ANDERSON R ET AL: "NetCard-a practical electronic-cash system" see page 49, paragraph 2 -----	4,11